RL-TR-95-292
Interim Report
February 1996

C3I KERNEL SELECTION REPORT

Honeywell, Inc.

Rakesh Jha

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

Rome Laboratory
Air Force Materiel Command
Rome, New York

This report has been reviewed by the Rome Laboratory Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be releasable to the general public, including foreign nations.

RL-TR-95- 292 has been reviewed and is approved for publication.

APPROVED: Richard C Metzger

RICHARD C. METZGER
Project Engineer

FOR THE COMMANDER: John Graniero

JOHN A. GRANIERO
Chief Scientist
Command, Control & Communications Directorate

If your address has changed or if you wish to be removed from the Rome Laboratory mailing list, or if the addressee is no longer employed by your organization, please notify Rome Laboratory/(C3CB), Rome NY 13441. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE February 1996	3. REPORT TYPE AND DATES COVERED Interim Oct 94 - Feb 95

4. TITLE AND SUBTITLE

C3I KERNEL SELECTION REPORT

5. FUNDING NUMBERS

C - F30602-94-C-0084
PE - 62702F
PR - 5581
TA - 20
WU - 78

6. AUTHOR(S)

Rakesh Jha

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Honeywell, Inc.
Honeywell Technology Center
3660 Technology Drive
Minneapolis MN 55418

8. PERFORMING ORGANIZATION REPORT NUMBER

N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Rome Laboratory/C3CB
525 Brooks Rd
Rome NY 13441-4505

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

RL-TR-95-292

11. SUPPLEMENTARY NOTES

Rome Laboratory Project Engineer: Richard C. Metzger/C3CB/(315) 330-7650

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

The main purpose of the benchmark suite is to provide an effective means to evaluate the use of high performance computing technologies for future C3I systems. This report summarizes the C3I kernels selected for inclusion in this benchmark set and the reasons for this inclusion. Each kernel is presented with an introduction and the associated parallelization issues.

14. SUBJECT TERMS

Parallel computing, C3I, Benchmarking

15. NUMBER OF PAGES

36

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL

TABLE OF CONTENTS

1 Introduction

The objective of this effort is to develop a suite of parallel benchmarks for the domain of C^3I (command, control, communications, and intelligence) applications. The benchmarks are being aimed at the evaluation of parallel hardware architectures as well as parallel software technology.

The benchmarking methodology will be based on the NAS (Numerical Aerodynamic Simulation) parallel benchmarks[1] and the parallel benchmarks developed by the PARKBENCH Committee[2]. The main reason for developing an altogether new set of benchmarks is that C^3I applications are significantly different from the scientific computations addressed by the existing benchmarks.

In this report, we discuss the benchmark selection criteria and present an outline of the selected benchmarks. Each benchmark outline describes the importance of the benchmark to C^3I systems, identifies the computations and input data required by the benchmark, and discusses the issues involved in parallelizing it. Ten out of the eleven kernels presented here will be developed further into complete benchmarks during the course of the program.

2 Benchmark Selection Criteria

The main purpose of the benchmark suite is to provide an effective means to evaluate the use of high performance computing technologies for future C^3I systems. To achieve this goal, care must be exercised in selecting individual benchmarks as well as in deciding which benchmarks will constitute the overall suite.

2.1 Selection Criteria for Individual Benchmarks

Each benchmark must represent the core computations in one or more important C^3I application areas. It should be computationally intensive and should have the potential for significant performance improvement through parallel execution.

The size of a benchmark should be carefully selected. A benchmark should not be so small that it fails to be representative of the computations and data movement in real applications. Similarly, it should not be so large that vendors and other implementors cannot justify the high cost of implementing it. The benchmarks developed under this program will consist of core computations of significant size (called *kernel* benchmarks) rather than complete end-to-end applications.

The input data set for each kernel should be large enough to allow at least two order of magnitude performance improvement. Ideally, multiple input data sets should be considered wherever that is necessary to expose significant transitions in parallel performance. Finally, the input data should be unclassified so as not to hinder wide dissemination of the benchmarks.

2.2 Selection Criteria for the Overall Suite

The members of the benchmark suite should complement each other. Together, the selected benchmarks should cover a wide range of algorithms, degree and granularity of parallelism, data partitioning, global versus local dependencies, interprocessor and intraprocessor data movement, computation-to-communication ratio etc.

3 Selected Kernels

Based on the criteria identified in the previous section, we have initially selected an initial set of eleven C^3I kernels from the list of twenty application areas suggested by Rome Laboratory.

1. Terrain Masking
2. Plot-Track Assignment
3. Synthetic Aperture Radar
4. Map-Image Correlation
5. Hypothesis Testing
6. Route Optimization
7. Discrete Event Simulation
8. Tracking
9. Threat Analysis
10. Image Understanding
11. Decision Support Systems

In our selection of kernels, we favored areas which involve computations that are unique to C^3I applications. This is the reason that we did not select the more general-purpose scientific computations such as 2-D transforms, convolutions, or ray-casting. These types of algorithms are well-studied in the scientific literature. Instead, we favored kernels arising out of more complete application functions such as route optimization, tracking, and synthetic aperture radar processing.

The following sections present an introductory description for each benchmark. Each section describes the importance of the benchmark to C^3I systems, and identifies the computations required by the kernel. We discuss the benchmark input data, and outline the parallelization issues associated with the benchmark. It should be noted that the kernel for Pattern Recognition has been dropped, and the kernel for Hypothesis Testing is different from the one presented at the program kickoff meeting. Of the eleven kernels presented in this report, ten kernels will be selected for further development in consultation with Rome Laboratory.

4 Terrain Masking

4.1 Importance to C^3I Systems

Terrain Masking computations are used in ground based C^3I, and in aircraft flight mission computer systems to aid in attack, covert, and evasive flight operations. When used in mission systems, evasive routes are generated which have low observability with respect to a set of given threats and their positions. Such threat positions may be known in advance and included in the database, or detected during flight.

4.2 Kernel Computations

A threat intervisibility calculation indicates what terrain surrounding a threat, e.g., anti-aircraft SAM site, is hidden from the threat sensor equipment, typically a radar-type sensor. Terrain may be hidden from the threat sensor equipment because it is either beyond the range capability of the threat, or is within range but terrain masked relative to the view of the threat installation. To properly calculate intervisibility, the terrain surrounding the threat must be fully analyzed, taking into account such factors as DTED database characteristics, radar range, threat elevation, earth curvature, and radar curvature along the earth's surface.

There are two basic approaches to computing intervisibility -- one based on radial scan of the area to be covered and the other based on a two-dimensional linear scan. Under the radial scan approach, data sampling of the DTED database occurs along radials emanating from the threat position, and having a length corresponding to the range of the threat. The basic algorithm computes the lower boundary of the detection envelope over the visited area, which may be represented in terms of the set of masking altitudes (h_j) at the visited data points j.

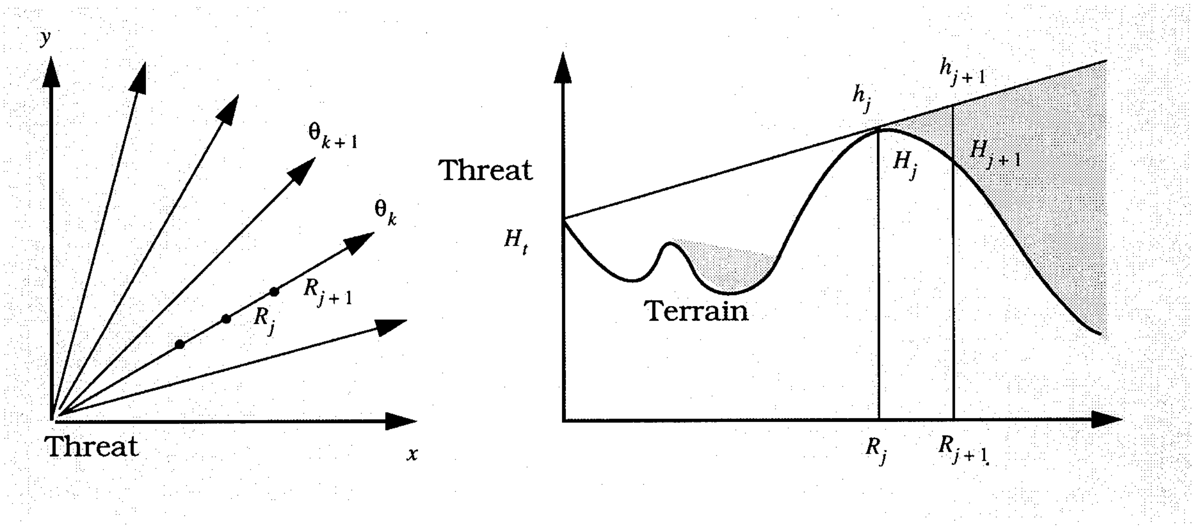

Figure 1. Terrain masking altitude calculation

Figure 1 shows a family of radials centered on a given threat and a cross-sectional view along a radial. As points j along a radial are visited, h_{j+1} may be expressed in terms of h_j by the equation

$$h_{j+1} = max\left\{H_{j+1}\ ,\quad h_j + \frac{1}{R_j}(h_j - H_t) + \frac{(\Delta R)^2}{2R_e}\right\}$$

Where H_t is the altitude of the threat, H_{j+1} is the altitude of the terrain at $j + 1$, R_j is the distance of the j^{th} data point from the threat, $\Delta R = R_{j+1} - R_j$, and R_e is earth's radius. If H_{j+1} is the greater of the two terms above, the terrain at the sample point $j + 1$ is visible to the threat, else it is masked. The shaded areas along the radials represent a volume of terrain masked space with respect to the threat.

As the visited points along radials do not always coincide with the DTED grid points, the terrain elevation at a sample point may have to be computed by interpolation of the DTED grid points in the immediate neighborhood of the sampled point. Similarly, the computed masking altitudes must be converted back by interpolation to the masking altitudes at the DTED grid points.

4.3 Parallelization Issues

The computational complexity of the terrain masking calculations depends on several operational parameters -- size of the gaming area, number of threats, ranges of threats, and the granularity of spacing between grid points. One obvious way to parallelize the kernel is to compute along different rays in parallel. For a coarse 5^0 angular spacing between rays and a 360^0 scan, there is potential for 72-fold improvement in performance. Where multiple threat are involved, each threat can be handled in parallel, which potentially results in total speedup of two orders of magnitude. In this case, the effective masking altitude at a point is the minimum of the masking altitude due to individual threats. As the computational load for a threat depends on threat parameters (e.g. range), the assignment of threats to processors will have to take load balancing into account.

4.4 Kernel Benchmark Data

The input datasets for the kernel benchmark will be obtained from the threat and DTED database from the Quiet Knight program. It will consist of terrain elevation data in earth-centered spherical coordinates with an angular quantization appropriate to the latitude of the area covered, unclassified threat data in terms of range and altitude, and typical values for other parameters such as size of the gaming area and desired granularity of sampling. An arbitrary placement of threats will be specified. Algorithms used in implementations must not depend on the specific placement specified in the benchmark.

5 Plot-Track Assignment

5.1 Importance to C^3I Systems

Plot-track assignment is at the heart of any surveillance and reconnaissance system which must infer the type and location of enemy forces in a cluttered electromagnetic or acoustic environment. This function is an essential step in fusing information from multiple sensors, since incoming measurement data must be correlated with existing track databases before the measurements can be used to improve the track accuracy and identification. The problem of plot-track assignment consists of associating the incoming sensor returns with the existing tracks, in a manner which maximizes the likelihood that the correct measurement-track matched pairs are obtained. Since sensors continue reporting new information periodically, it is imperative that the data reports from the sensors be correlated in real-time. Currently, this imposes a major limitation on C^3I systems, as processing requirements can quickly exceed the available time between sensor reports.

5.2 Kernel Computations

Our kernel in plot-track assignment represents a typical mixture of the operations in operational algorithms. Figure 2 illustrates the major computational blocks in the proposed kernel. A set of tracks is available to start the algorithm, where each track is represented in earth-centered inertial coordinates by their estimated 3-dimensional positions and velocities and the associated 6 x 6 covariance uncertainty matrix. The number of tracks can be scaled to provide opportunities for additional parallelism; at this point, we expect to provide two data sets: a small benchmark consisting of 50 tracks, used primarily for program development and debugging, and a larger benchmark consisting of nearly 5000 tracks.

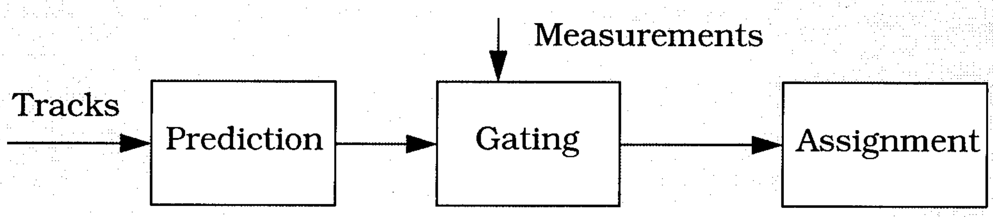

Figure 2. Major computation steps in Plot-Track assignment.

The purpose of the plot track assignment algorithm is to correlate the existing track database with a returned measurement set from a single sensor. The measurement set is collected at a time different from that of the track database, and contains missed detections and spurious returns which do not correspond to any tracks. For this kernel, we propose that the measurement set consist of noise-corrupted two-dimensional projections of the true object positions onto a sensor reference position. Thus, each measurement will consist of an azimuth and an elevation angle, relative to a sensor-oriented coordinate system. These measurements correspond to those collected by a passive long-wave infrared telescope of limited accuracy.

The computations required in the plot-track assignment algorithm are as summarized as follows:

- Prediction: Extrapolate the entire track database to the time of the measurement set, including the covariance, and convert the extrapolated positions into predicted angular measurements in sensor coordinates
- Gating: Compare the predicted angular measurements to the actual measurements, and compute a likelihood of assignment based on the covariance accuracies. Based on this likelihood, identify as a minimum all candidate matches between predicted and actual measurements with a likelihood exceeding a small threshold.
- Assignment: Select the most likely assignment of predicted tracks to measurements in the candidate assignment set using the optimal assignment algorithm of Jonker and Volgenant, as modified by Castañon [3].

In our detailed specification, we will provide specific numerical algorithms for prediction. However, we will leave the choice of gating algorithm as an implementation design. In particular, the reason for using gating is to reduce the number of candidate assignments to be considered in the assignment algorithm; in architectures which do not effectively exploit sparse data structures (such as SIMD architectures), it may be more efficient to skip most of the Gating operation. For the final step, we will provide a specification for the JVC algorithm in [3]. This algorithm is an efficient variation of an optimal primal-dual assignment algorithm, which has proven to be among the most efficient sequential algorithms for plot-track correlation.

5.3 Kernel Benchmark Data

The kernel benchmark data will be derived from the IDA 6260 targets scenario used extensively in previous benchmarks by the multi-object tracking community [4]. An initial timing of a plot-track correlation benchmark code required over 100 seconds of computation time on a Solbourne Sparcstation (25 Mips) using this scenario. To make the kernel sufficiently rich, we propose to use a larger scenario based on a similar geometry. The resulting data set will capture accurately the algorithmic aspects of plot-track correlation in modern C^3I systems.

In order to verify correctness of kernel implementations, two primary outputs will be checked: The number corresponding to the optimal likelihood of the matched plot-track assignments, and the assignments which form the optimal plot-track assignments. Timed computations will start once the track database and measurement sets are read in, and will continue until the optimal assignment is found; thus, I/O time will not be included in the timings.

5.4 Parallelization Issues

The three main algorithmic steps have different parallelization potential; the challenge is to combine the parallelization of the three steps to achieve an overall efficient algorithm. Our past experience in programs such as [5,6,7,8,9] and in our research publications [10,11,12,13,14] suggests that speedup of over 100 is possible on some architectures.

The first computational block, Prediction, is easily parallelized on a per-target basis. The second computational block, Gating, is an associative operation which can also be parallelized on a per-target basis, or on a per-measurement basis. Furthermore, clever data structures such as the sort-based gating scheme used in SIMD architectures in [9] can often increase the parallelization effectiveness of the gating procedure. The parallelization bottleneck is the optimal assignment algorithm, which is roughly 5% of the overall computation time. We estimate based on previous research results that the parallel speedup potential of this block is under 10, so the key challenge in this kernel will be to effectively combine parallelization across the three steps.

6 Synthetic Aperture Radar

6.1 Importance to C^3I Systems

Automatic Target Recognition (ATR) is a critical problem in today's battlefield. Synthetic Aperture Radar (SAR) imaging is one of the principal surveillance sensors for ATR, due to its ability to penetrate sophisticated camouflage. Low resolution SAR sensors are a current part of the JSTARS sensor suite, and high-resolution SAR sensors form the basis of many experimental ATR systems being developed under ARPA sponsorship. In SAR imaging, 2-D Fourier transforms and inverse 2-D Fourier transforms are used to reconstruct the high-resolution range-azimuth scattering images from the space-time indexed signal returns collected at the radar. However, SAR processing requires interpolation from the sensor-based coordinate system to spatial coordinates, thus creating a broad mix of computations.

6.2 Kernel Computations

Our proposed computational kernel in SAR processing is based on the application of spotlight SAR imaging. SAR imaging is an inherently 2-dimensional process; the raw data collected by the sensor is indexed by the location of the sensor along the flight path, u, and the time at which it is collected, t. The basic steps in the SAR imaging algorithm can be summarized as follows [16] --

- Convert the received signal to baseband, resolving it into in-phase and quadrature (I/Q) components to support phase-coherent processing of the signal.
- Perform range compression. In most SAR systems, the transmitted signal is a linear FM pulse. After "deramping", the received signal is a frequency transform of the range history. Recovering the range information in the received signal is accomplished by means of a DFT. At the same time, ancillary processing such as range equalization and side-lobe suppression is performed.
- Perform azimuth compression. Azimuth compression is the operation that converts the set of range compressed pulses to a two dimensional image in range and cross-range coordinates. Azimuth compression is accomplished by convolving the received signal in the u dimension with respect to a convolution kernel that is a function of the range variable. The convolution for each range bin is accomplished by a DFT followed by multiplication by the transform of the convolution kernel and an inverse DFT.

Figure 3 below depicts the processing outlined above for the SAR processing benchmark

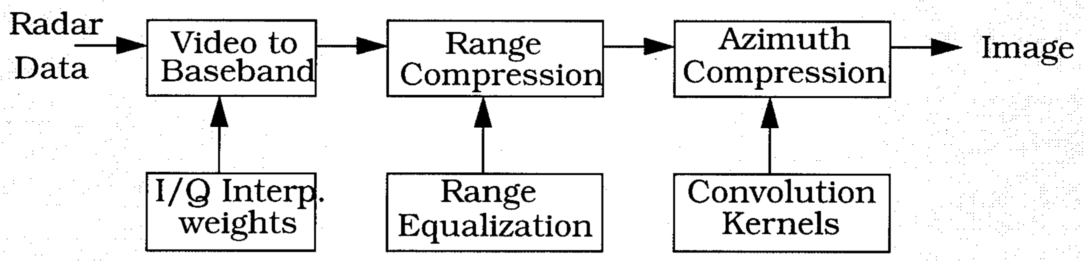

Figure 3. SAR Kernel Processing

The SAR kernel is based on the SAR imaging algorithms described in [16], and used in conjunction with Lincoln Lab's Advanced Detection Technology Sensor (ADTS) [15].

6.3 Parallelization Issues

Due to the presence of the Fourier transform and interpolation operations, SAR processing can be computationally intensive for generating high-resolution images over large areas. As a calibration point, SAR processing times of over 100 minutes for 4096 x 4096 images on a T800 transputer (15 MIPS, 2.25 Mflops) were reported in [17]. Such delays would be unacceptable in the processing of real-time information for C^3I. There are several studies on parallelization of transform operations [18, 19, 17] which highlight the extensive parallelizability of Fourier transform-like operations. These studies indicate that using parallel computing on SAR processing for high-resolution images can result in speedup of several orders of magnitude, provided that the implementations can avoid excessive data movement between processors in order to switch from transform operations to interpolation operations. The SAR benchmark algorithm described here is a "rectangular" algorithm, and does not have significant 2-D interpolation operations, which are characteristic of "polar" algorithms associated with spotlight mode SAR.

6.4 Benchmark Data Sets

In order to have readily-available unclassified data, the SAR processing kernel is based on the SAR imaging algorithms used in conjunction with Lincoln Lab's ADTS [15, 16]. This system has been used to obtain numerous data sets of SAR signals and images of terrain scenes, from areas such as Stockbridge, NY and Presque Isle, ME. A portion of the Stockbridge data is available from Lincoln Laboratory as part of the ADTS program [20]. The data provided in [20] comprises processed images of the Air Calibration Site, together with calibration information. In addition to this image data set, the ADTS system records the radar video data prior to image formation. Subject to the ADTS program sponsor approval, the video data can also be made available. These data sets will provide realistic, unclassified inputs to be used in benchmarking.

The ADTS transmits linear FM pulses with alternating (H and V) polarizations and simultaneously receives both H and V polarizations. The recorded video data consists of 2032 even/odd pairs of 11 bit data words per pulse for each of the four transmit/ receive polarizations, HH, HV, VH, and VV. The benchmark SAR algorithm forms images from 512 same-polarization pulses, yielding a 2048 (range) by 512 (cross range) point image.

Validation of the parallel implementations of the SAR benchmark is facilitated by the benchmark data set in two ways: First, there are output images available, making it feasible to compare the output of the parallel implementation with the "correct" image. Suitable tolerances allowable errors in the output image (peak and root mean square errors) will be developed that allow for roundoff or truncation errors will be developed as part of this program. Second, the Air Calibration Site data contains a set of reflectors with known positions, allowing an implementation to be partially validated without recourse to an output data set. This technique would be useful in initial development and testing.

7 Map Image Correlation

7.1 Importance to C^3I Systems

Modern C^3 I systems often include the capability to accept surveillance data from remote imaging sensors such as space-based infrared satellites, remotely-piloted vehicles (RPVs) or intelligence photographs. In order to extract useful information from these sensors, it is important to determine accurately the alignment of features in the images with a detailed map of the area of interest. If GPS-aided navigation systems are used, navigation system uncertainties can be reduced significantly. However, for long slant-range surveillance, the errors in the estimate for platform orientation make automated registration from sensor imagery to digital map difficult. For example, for a surveillance range of 180 km., which is the field of regard of the JSTARS low-resolution SAR sensor in the direction perpendicular to the line of flight, an orientation error of 0.1 degree causes a misregistration of 300 m. Because the sampling density of Digital Terrain Elevation Data (DTED) and Digital Feature Analysis Data (DFAD) is 25 m., the estimated location of landmarks across the field of regard will be off by several bins.

7.2 Kernel Computations

The kernel is based on a scheme for establishing the registration of a SAR image to the map coordinate system via a two-stage feature matching process. In it, the predicted locations of digital map features (given an initial estimate of the platform's location in the map) are compared with the actual locations of features extracted from a narrow field-of-view SAR image. In the first stage, coarse features (i.e. cities or bodies of water) in the map are used to provide an initial estimate of the map-to-image alignment. Once this information is available, the second stage uses more precise features (such as road intersections or rivers) to achieve a high degree of registration accuracy. This approach effectively combines the position and attitude information provided by an on-board inertial navigation system (INS) with the precise location information provided by the map-to-image registration process.

The proposed kernel computations will be the first stage of the map-image correlation process, namely, coarse feature match. In particular, the identified kernel will be based on the presence of cities as the SAR image features, since they are easy to locate due to the large number of corner reflectors (dihedral angles) that provide strong returns in the image. Once the city regions have been extracted from the SAR image, the map-image correlation kernel will match them against the collection of city regions obtained from the digital map. Candidate city pair matches are identified on the basis of city size and distance between pairs of cities in the map and the image. Figure 4 shows the functional block diagram of the kernel.

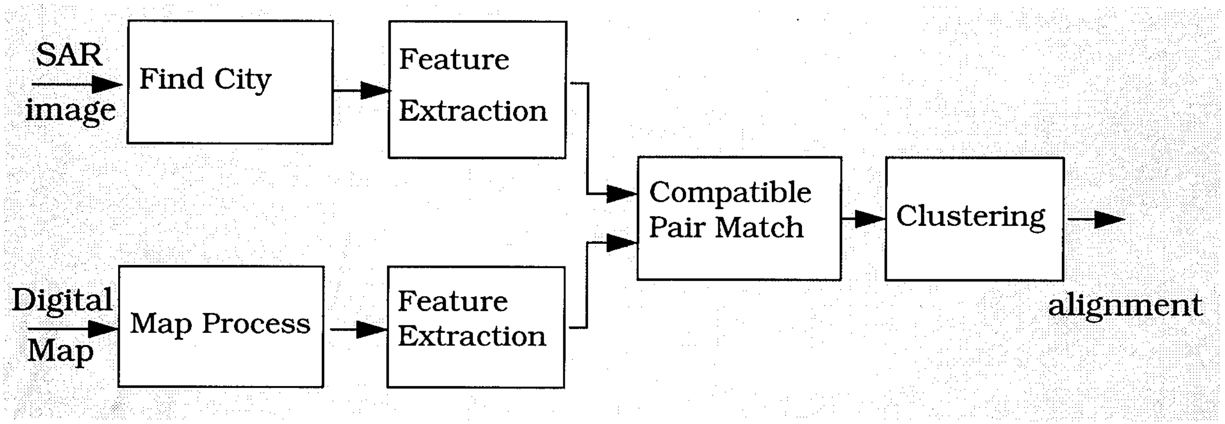

Figure 4. Functional block diagram of Map-Image correlation

The Find City block locates city regions in SAR image by identifying groups of bright pixels in the image using an amplitude screening operator. These results are then "smeared" using a morphological closing operator. Following the closing operator, the intermediate results are then opened and closed again to isolate the large regions in the image which correspond to the possible city locations.

The Map Process block locates city, town and village regions in a digital map and produces a new binary image that contains only those regions. The Extract block computes the size and x- and y- centroid of each city region in the input image. The sizes of the cities are scaled to correspond to the sizes of the cities in the SAR image. A threshold can also be used to eliminate small city regions.

The Compatible Pair Match block generates the possible line segment combinations by comparing the sizes of the regions. A pair of size features in the SAR image compares with every pair of size features in the Map to determine its compatibility. Lastly, the Clustering block groups the line segment combination in order to determine the best set of matching segments.

7.3 Parallelization Issues

As shown in Figure 4, the SAR and digital map can be processed in parallel up to the compatible Pair Match block; one can apply task parallel methodologies to speed up this portion of benchmark. Furthermore, the Find City block can be executed in a data parallel manner by partitioning the SAR image. Similarly, the Map process can also be executed in a data parallel manner.

Feature extraction computations for the SAR image and the digital map are basically the same, and may be executed using data parallelism. The complexity of the matching calculation can rapidly become unwieldy as the number of feature pairs grows because for every pair of features in the image, the set of map features must be searched to find compatible matches. This computation can be performed independently for each pair of features, and the results combined across processors to obtain clusters and identify the most likely matching.

These decomposability indicates that performance speedup of at least two orders of magnitude should be possible for large images with many features. Further speedup may be possible depending on the hardware and software tools.

7.4 Kernel Benchmark Data

We have identified and located a database of unclassified SAR imagery from West Germany. This data is for a 35x35 km. ground footprint. We selected portion of this data for benchmark evaluation. The sizes of the selected SAR images are 512 x 512, 512 x 1024, 512 x 1536, 1024 x 512, and 1536 x 512 pixels. Co-located standard DTED I and DFAD I with 100 meter post spacing is also available for the terrain regions viewed by the SAR sensor. The size of the digital map data is 1280 x 1280 pixels.

8 Hypothesis Testing

8.1 Importance to C^3I Systems

Hypothesis testing is one of the four basic steps in Wohl's classic Stimulus Hypothesis

Option Response (SHOR) [23] paradigm of Command and Control. In this paradigm, stimulus represents the processed sensor information; this stimulus is used to generate and validate hypotheses concerning the tactical situation. Broadly interpreted, hypothesis testing algorithms are used throughout C^3I, in areas such as object classification, discrimination, multi-object tracking using multiple hypothesis methods, and kill or damage assessment. What distinguishes hypothesis testing in tracking from that in pattern recognition or other classification problems is that for tracking it is a temporal real-time process, where additional information is collected over time to help improve the assessments of hypotheses.

Our proposed benchmark kernel for hypothesis testing is based on tracking of maneuvering targets based on radar measurements. In this problem, a nominal model of target motion is available for many possible maneuvers; during each time interval, a target chooses one of the possible maneuvers to execute. The objective of the hypothesis testing algorithm is to determine the sequence of maneuvers (and thus track the target accurately) based on the noisy radar measurements. Similar hypothesis testing problems arise in speech recognition, where a sequence of utterances must be recognized based on noisy measurements of the individual sounds.

Note that this kernel is different from that used in the original proposal. Our decision to change the kernel was based on two reasons. The original proposal called for using an optimal branch-and-bound hypothesis testing algorithm, based on track initiation in clutter. We felt that using an optimal NP-complete algorithm would make a poor parallel benchmark, since speedup in such problems is often a function of the order in which nodes are explored, and this order is hard to maintain in the specification without handicapping parallel architectures. Such objections would be overcome by replacing the branch and bound algorithm with a heuristic with more predictable computations; however, the kernel would then resemble closely the tracking kernel. Thus, we chose to change the nature of the kernel.

8.2 Kernel Approach

The approach proposed for this kernel is that of using an optimal maximum likelihood estimation problem for what is known as a hybrid model. The basic maneuvering target tracking model is described mathematically as:

$$\begin{aligned} x(t+1) &= A(q(t))x(t) + w(t) + d(q(t)) \\ y(t) &= Cx(t) + v(t) \end{aligned}$$

where $x(t)$ is the state representing position and velocity, $q(t)$ is an index denoting the maneuver index at stage t, $w(t)$ and $v(t)$ are white noise processes representing modeling uncertainty and measurement uncertainty, respectively, and $y(t)$ are the radar outputs at time t, assumed to be noisy estimates of object position. The input $d(q(t))$ is the acceleration which is associated with a given maneuver $q(t)$, and the matrix $A(q(t))$ denotes how the choice of maneuver affects the state dynamics.

The maneuver process $q(t)$ is modeled as a finite-state Markov chain, with a specified set of probability transitions. The nature of the maximum likelihood hypothesis testing problem is to determine the most likely sequence $q(0), q(1), \ldots, q(T-1)$, based on the above models and the observation of the measurements $y(0), y(1), \ldots, y(T)$. Assume that there are 10 maneuver types which can be used at each stage. Then, the set of all possible maneuver sequences over T transitions is 10^T. Thus, the set of possible hypotheses grows exponentially!

Our proposed benchmark kernel computations are illustrated in Figure 5. At each stage t the algorithm starts with a set of candidate hypotheses, concerning the previous sequence of maneuvers $q(0), \ldots, q(t-1)$, and the associated likelihood that each hypothesis is correct. At stage t, the measurement $y(t)$ is obtained, and a new set of possible sequences is generated and evaluated, concerning maneuver sequences $q(0), \ldots, q(t)$. In order to keep the computation complexity under control, a gating algorithm can be used so that only the 1000 most likely maneuver sequences are kept for the next interval; this requires comparison of the likelihoods of all the maneuver sequences, and selecting the 1000 most likely ones. The selection of 1000 alternative hypothesis should provide enough opportunity for parallel speedup, particularly since each of these hypothesis can have up to 10 continuations which must be evaluated.

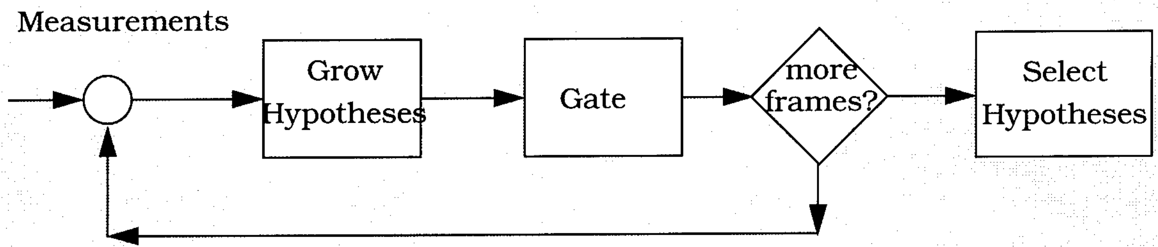

Figure 5. Hypothesis Testing Kernel.

8.3 Kernel Benchmark Data

In order to keep the kernel simple, we propose to use a 2-dimensional (constant altitude) maneuvering aircraft model, with 10 levels of maneuvers. The data will be generated using a random problem generator which will select a sequence of maneuvers. Thus, the input data for this problem will consist of 8 pairs of $\{x, y\}$ positions, together with the specifications of the matrices $A(q)$, C, and the vectors $d(q)$ for up to 10 maneuvers, and with the specification of the transition probabilities of the Markov chain governing q.

Verification of the kernel will be based on the numerical likelihoods computed for a subset of maneuver sequences which must be in the top 1000 sequences evaluated, as well as the likelihood identified for the most likely maneuver sequence.

8.4 Parallelization Issues

This kernel has a nearly linear speedup potential, scalable to large numbers of processors. The key issues are designing parallel schemes for generating the hypotheses, a parallel gating algorithm for selecting the 1000 most likely sequences, and a load-balancing algorithm for redistributing the computational load after the 1000 most likely sequences are selected.

9 Route Optimization

9.1 Importance to C^{3}I Systems

Route optimization is an important subproblem which arises in the Air Battle Planning Process and Air Tasking Order generation. Once targets are selected, and appropriate aircraft and weapons are matched to the targets, route optimization algorithms are used to find flight paths which increase the likelihood that the aircraft will accomplish their missions. Route optimization must consider the capabilities and locations of available enemy surveillance sensors and anti-aircraft weaponry, and exploit potential terrain masking to design routes with minimal potential for enemy interference. Route optimization must also consider the effect of supporting resources (such as standoff jammers) on enemy surveillance capabilities. These problems also arise in other C^{3}I environments such as troop movements of ground forces.

9.2 Kernel Computations

The proposed kernel in route optimization is based on finding the best route from a specified origin location to a number of destinations. In the Air Force context, best is often interpreted in terms of minimum risk; this type of route optimization is often used in automated systems for mission planning, such as the Force-level Automated Planning System (FLAPS) [21].

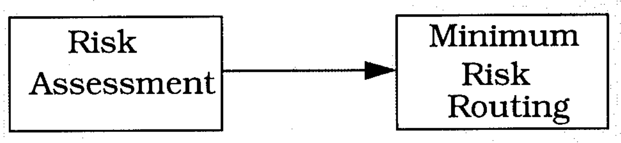

Figure 6. Principal Modules in Route Optimization

The inputs to the route optimization algorithms are a discrete space of 3-dimensional cells, together with an analytical measure of the risk associated in traveling from one cell to another. This risk is computed as a function of the distance and geometry from the cells to a set of threat locations, and is a function of visibility and maneuverability.

The principal computations in the route optimization kernel are outlined in Figure 6. The first block, risk assessment, must evaluate the risk associated with each cell as a function of the geometric computations between the cell and the threat locations. One of the components of the risk function is the terrain masking output. In the second block, a minimum risk route is found using a shortest path algorithm. We propose to leave the choice of shortest path algorithm as a design parameter in the benchmark kernel computations; we will provide a specification for an efficient sequential algorithm candidate, based on the Bellman-Ford variation of the dynamic programming algorithm.

9.3 Kernel Benchmark Data

The input for this kernel will be in terms of a matrix that contains the vulnerability index for each cell in the matrix. The matrix represents the physical area over which the route optimization is to be done. The vulnerability index is a measure of the risk associated with traversing a cell. One of its components may be the terrain masking altitude at a cell for a given set of threats.

Test cases for Route Optimization will be selected from Honeywell's work on Threat Avoidance Analysis on the Quiet Knight program [22]. Although many of these scenarios involve terrain and enemy capability models which may be classified, unclassified scenarios will be generated based on hypothetical scenarios.

9.4 Parallelization Issues

The principal parallelization issue in the choice of our kernel is to develop a parallel shortest path algorithm which can achieve two orders of magnitude speedup when compared to efficient sequential shortest path algorithms. Fast shortest path algorithms are usually incremental, and do not parallelize well. However, the geometric structure of the route-optimization problem allows for efficient parallel dynamic programming algorithms, such as the Bellman-Ford algorithm. Given the large size of the problem space, we expect that this algorithm will be parallelizable, and have the potential for up to two orders of magnitude speedup.

10 Discrete Event Simulation

10.1 Importance to C^{3}I Systems

Among the most critical issues in the analysis of C^{3}I systems is a proper understanding of the delays associated with information acquisition and dissemination. From an understanding of the delays in enemy systems, one can identify weaknesses, and design attack routes which exploit these delays [24]. Discrete event simulation is the method of choice for conducting these analyses; however, modern sequential computers are too slow to allow the use of real-time simulations in C^{3}I systems. The use of parallel processing technology opens up the possibility of developing decision aids which include real-time discrete event simulations.

10.2 Kernel Computations

The proposed kernel is a discrete event simulation engine that accepts inputs in the form of a class of timed, Stochastic Petri Net (STPN), similar to ALPHATECH's Modeler algorithms [25]. STPNs involve tokens flowing through a bipartite, directed network consisting of places and transitions. Places store tokens, unchanged, while they await certain conditions to hold at downstream transitions. Transitions fire when - 1) at least one token exists in each upstream place, and 2) some Boolean enabling function evaluates as true when applied to a set of tokens, one from each input place. Upon firing, transitions may remove a token from its input place, and create a new token in each output place. The time a transition fires may depend on time and random elements as well. STPNs have been used to model a broad range of C^{3}I systems such as air defense command and control for the Air Force's Foreign Technology Division (FTD) [24], air tasking order generation [26] and mobile surveillance systems. The Petri net structure is flexible enough to represent complex simulations, but is also explicitly focused on the event structure, so that low-fidelity models can be used to reduce the kernel's software complexity.

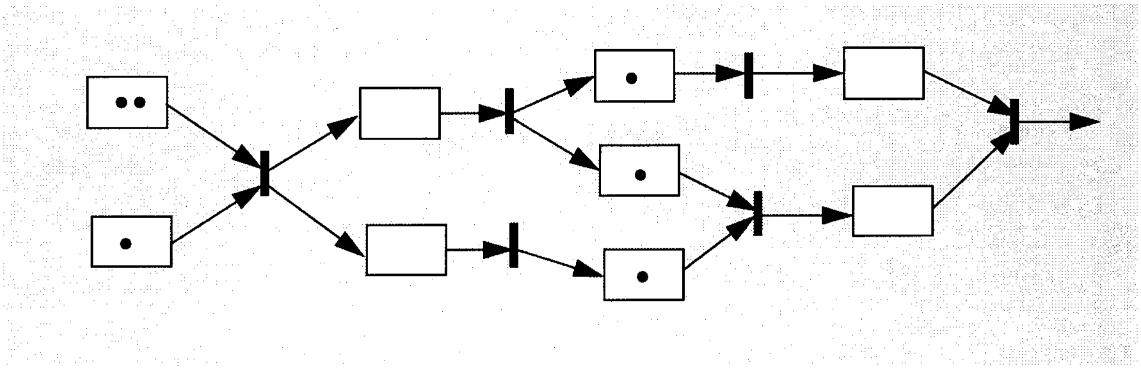

Figure 7. Illustration of Stochastic Timed Petri Net Model.

The general class of Petri nets is too broad, and can result in models which are difficult to parallelize due to the extensive synchronization requirements. We propose to generate a model corresponding to a subclass of Petri nets known as Marked Graphs. Marked graphs have the property that only one transition can follow each place. Thus, conflicts between transition timings arising from two places accessing the same token are avoided. Each transition can fire independently of other transitions as soon as all of its inputs are enabled. Firing a transition involves generation of a random variable corresponding to the transition time, and eventually generating a token to be placed in each of the output places.

10.3 Kernel Benchmark Data

The Honeywell team has developed many unclassified discrete-event simulations of C^3I systems in previous programs [26,24]; however, the data for these simulations is unnecessarily cumbersome, due to the detail required to model an operational system. We propose to generate a sizable model of a marked graph, using the hierarchical tools in the Modeler application [25]. This benchmark data would be based on a simplification of ALPHATECH's Air Tasking Order generation model documented in [26].

The input will consist of the model structure, in terms of a graph connecting places and transitions, a random number generator and a set of initial seeds for each transition for computing the transition times, and an initial marking of the net. Validation of the implementation can be based on the transition time to reach a final marking, and the final marking reached.

10.4 Parallelization Issues

The main challenge in parallelization of the marked graph model is in efficiently monitoring and determining which transitions must fire. The marked graph nature of the structure allows for a simple, asynchronous implementation of the transition rule, as each transition can be monitoring its input places in parallel. Note that there is a significant amount of communications and load balancing problems which must be solved in order to achieve efficient parallelization.

Since discrete-event simulations require the generation of large numbers of random variables, we will enforce consistency between parallel and sequential computations by using identical seeds and random number generators for each transition. In this manner, the set of random numbers selected will be uniquely specified.

11 Tracking

11.1 Importance to C^3I Systems

Multi-target tracking is the primary real-time surveillance function of a C^3I system. Data is reported periodically by one or more sensors; this data must be correlated with existing track databases, in order to refine the track and identification accuracies through information fusion, and to establish whether new tracks are present. Multi-target tracking of cruise missiles using data from different sensors was the focus of ALPHATECH's Multi-sensor, Multispectral Fusion Program [27] sponsored by Rome Lab. Multisensor, multi-target tracking of large numbers of midcourse objects was the principal challenge involved in designing a practical Ballistic Missile Defense system for large-scale attacks [4]. In the MISTC and A2 programs, sponsored respectively by the Army's Strategic Defense Command, and in Rome Lab's Battle Management Structures program, the Honeywell team made significant progress in demonstrating the applicability of parallel processing to Ballistic Missile Defense tracking.

11.2 Kernel Computations

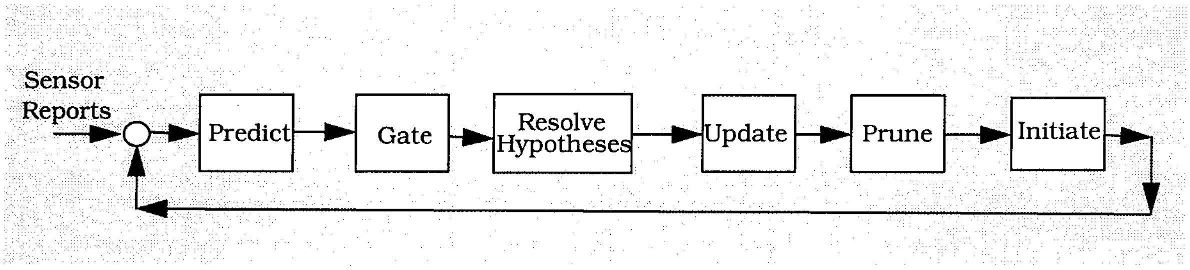

Figure 8. Illustration of Tracking Algorithm Computations.

The typical steps in a tracking algorithm are [28] -

- Receive Sensor Reports: Data collected by a sensor corresponding to a single scan of the scenario is provided to the tracking algorithm.
- Predict Tracks: Tracks in the database are extrapolated to the time of the sensor reports, and converted to sensor measurement coordinates to form predicted measurements. In addition, the accuracies associated with those tracks are extrapolated and converted to measurement coordinates.
- Gate: All measurements are compared with all predicted tracks to determine credible track-measurement associations. The resulting candidate associations are hypotheses, which must be resolved.
- Resolve Hypotheses: The likelihood of candidate hypotheses generated by either the current gating process or past gating processes is evaluated. Using an approximate maximum likelihood algorithm, a subset of the track hypotheses is selected for future resolution. These track hypotheses are required to be compatible in

that prior measurements are associated to only one track, thus removing the ambiguity created in gating.

- Update Tracks: Each track hypothesis represents a particular association of the measurements to tracks; thus, for each hypothesis, the information provided by the recent measurement is fused with the track information to obtain refined track information. This operation is often performed by an Extended Kalman Filter.
- Prune Hypotheses: Hypotheses which are no longer considered active after resolution are removed, so that new ones can be created in the next sensor scan.
- Initialize Tracks: Sensor measurements which are not associated with existing tracks are used to generate new track hypotheses for evaluation based on future measurements.

The above description highlights the cyclic, data-driven nature of tracking algorithms. Due to the real-time processing requirements, it is essential to use efficient parallelization in order to handle large-scale modern scenarios with multiple sensors and thousands of tracks of interest. Tracking also poses an interesting mix of floating point, easily parallelized computations (e.g. predict tracks, update tracks), data movement and distribution (receive sensor reports, gating), and combinatorial processing (resolve hypotheses).

Our proposed benchmark kernel is based on what is known as multiple-hypothesis tracking algorithms. The key difference between tracking algorithms is the approach used to resolve hypotheses. In 0-backscan approaches, hypotheses concerning the association of sensor measurements to tracks are resolved in the same cycle that the measurements are received, using optimal assignment algorithms such as the JVC algorithm [3], or suboptimal heuristics such as nearest neighbor or row-column algorithms [9]. In contrast, n-backscan approaches resolve hypotheses concerning the association of sensor measurements to tracks after n additional sensor reports have been incorporated in to the hypotheses. Such algorithms are known as multiple hypothesis tracking algorithms, since, for each candidate track, the database maintains a tree of possible hypotheses which correspond to that track, and which will be resolved when future sensor measurements are available.

In order to simplify the benchmark kernel computations, we will eliminate the track initialization aspect of the algorithm, and focus primarily on the track continuation computations. Our kernel corresponds to estimating the 3-dimensional position and velocity of targets undergoing simple, linear motion with random accelerations, using data obtained from radars which report errored 3-dimensional positions. In addition, the radar reports include random Poisson clutter, with uniform density over the radar coverage. The tracking algorithm will be a 2-backscan multiple hypothesis algorithm. The suboptimal Resolve Hypotheses algorithm will be a greedy algorithm, which must first find the most likely single track hypotheses and start a partial hypotheses set. Subsequently, the most likely hypothesis which is compatible with all of the partial hypotheses already in the set is added to the set. The algorithm stops when no compatible hypotheses can be found. The partial hypotheses set is used to prune all branches in the track trees 2 scans back which do not contain a member of the hypotheses set.

In order to allow tuning the algorithm to specific parallel architectures, several steps in the tracking algorithm will be specified at a functional level. These steps, such as gating and hypotheses resolution, can exploit the use of special data structures which differ among architectures.

Validation of the implementation will be based on the set of hypotheses which are maintained by the algorithm after processing of 30 seconds worth of data. Double precision computations will be required for the real-valued likelihood, prediction and update computations based on Kalman filtering and model identification algorithms.

11.3 Benchmark Data

As part of our previous work, we have access to large-scale unclassified scenarios. We propose to use 30 seconds worth of data from the IDA 6260 scenario [4] discussed previously, with three radar sensors reporting scan data every 10 seconds. Thus, the kernel will require the tracking algorithm to process data dynamically across several scans. The resulting data set will provide a realistic kernel, of sufficient size to evaluate accurately the various parallel performance metrics of interest.

11.4 Parallelization Issues

Our previous experience in the design and implementation of sequential and parallel multiple hypothesis tracking algorithms [27,7] on SIMD [9] and MIMD [6,7,8] architectures has shown that speedups of several orders of magnitude are possible; however, special data structures and algorithm variations must be used to exploit the features of each architecture. For this reason, our proposed kernel will allow variations in algorithm data structures across architectures, and thus provide a challenging and realistic benchmark.

12 Threat Analysis

12.1 Importance to C^3I Systems

In C^3I systems, Threat Analysis is used to interpret the surveillance and intelligence information collected, and to provide the human Command and Control with an assessment of the potential enemy actions, as well as the available options which can be used to negate each threat. Often, this function is performed by the Command and Control staff, based on track summary information and intelligence reports. Nevertheless, there are several aspects of this function which are automated; for threats under track, there is a need to determine in real-time which interceptor options can be used to destroy these threats, and the windows of opportunity for these options (threat accessibility). These operations must be executed as fast as possible, so that valuable intercept actions are not wasted due to processing delays.

12.2 Kernel Computations

Our proposed computation kernel in Threat Analysis is based on an abstraction of algorithms developed by the Honeywell team for theater missile defense applications [29]. The inputs consist of a set of threat tracks with uncertain positions and velocities, and a set of interceptor bases with different capabilities (abstractions of assets such as air fields, Patriot Batteries, and THAAD interceptors).

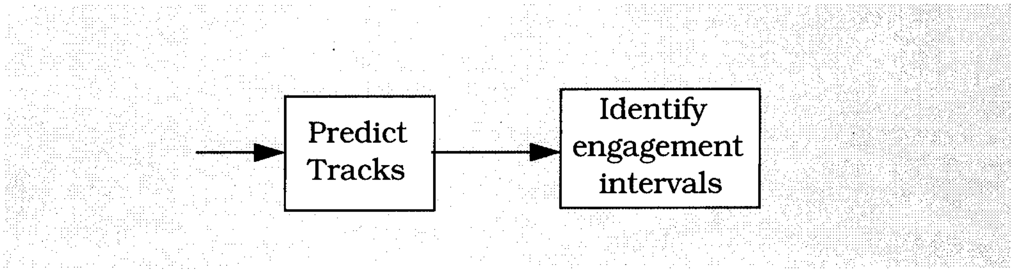

Figure 9. Illustration of Threat Analysis Computations

The threat analysis algorithms must extrapolate trajectories forward in time using ballistic trajectory equations. For each extrapolated position, the feasible trajectories of interceptors from each of the possible launch farms is checked to determine the earliest launch time and latest launch time (and associated intercept times) for each launch site against each incoming threat trajectory.

12.3 Kernel Benchmark Data

For this kernel, we propose to use test data generated from an unclassified theater missile defense scenario in the Middle East. This test data will consist of a set of threat state vectors, generated shortly after impact, and the location of a suite of different interceptor batteries, corresponding to capabilities similar to Patriot and THAAD batteries. Parametrized flyout templates will also be available, in the form of flyout templates for both Patriot and THAAD.

The primary output of the threat analysis algorithm will be a set of time windows, indexed by both battery index and threat index, which will represent the times during which a particular battery can intercept a particular threat. The start and end times for these windows will be restricted to integer values of time. Thus, verification of correctness will be based on the values of these windows.

It is possible to generate a set of input scenarios which will scale with the number of threats and the number of batteries. As part of the kernel specification, we propose to generate such a nested set of scenarios so that appropriate determination of measures such as speedup and isoefficiency is possible.

12.4 Parallelization Issues

We expect that this kernel will result in large speedup, since each threat can be processed independently. A more aggressive approach would be to process in parallel threat-battery combinations. Furthermore, there are opportunities for effective use of spatial data structures such as quadtrees or monotonic Lagrangian grids [30,9,31] in order to achieve additional speedups.

13 Image Understanding

13.1 Importance to C^3I Systems

Image Understanding (IU) is important to C^3I systems due to the information which it can autonomously provide regarding the contents of the scenes/environment imaged by the sensor(s). IU is a broad category of vision algorithms and applications. Some of the obviously useful IU algorithms include object recognition and image segmentation, which are key components in obtaining the location and configuration of enemy installations and equipment. Such information is invaluable in directing resources and assets in the course of a conflict.

13.2 Image Understanding Process

The IU technique of Model-Based Vision (MBV) has been successfully employed in the task of recognizing objects such as tanks, trucks, and buildings. MBV typically employs three fundamental steps in its operation: feature extraction, indexing, and verification. The steps are typically very time consuming in all but the most trivial applications. For example, the extraction of features can be quite intensive depending upon the type and number of features employed (e.g., segmented regions, corner points, edge contours, lines fit to edges, etc.). Indexing involves the matching of the extracted features to multiple models of objects where each object has multiple views from which it can be viewed (and hence, from which it needs to be matched). The process of indexing can produce multiple object hypotheses which must be fully checked/verified against the sensor imagery for consistency (i.e., there must be a rigid transformation which will make the object model be in agreement with the sensor image). The verification step involves making a decision on the object class, or identifying the need for additional features and deferring the object class decision.

13.3 Kernel Approach

The selected kernel is for feature extraction of model-based object recognition, which is the first critical, fundamental step in image understanding. The kernel algorithm is called Texture Boundary Locator, which detects and extracts features from the regions of interests based on the texture of the regions. The kernel algorithm works best on an optical (visible spectral band) image since it has strong textural information. SAR image is also a good input candidate for this kernel.

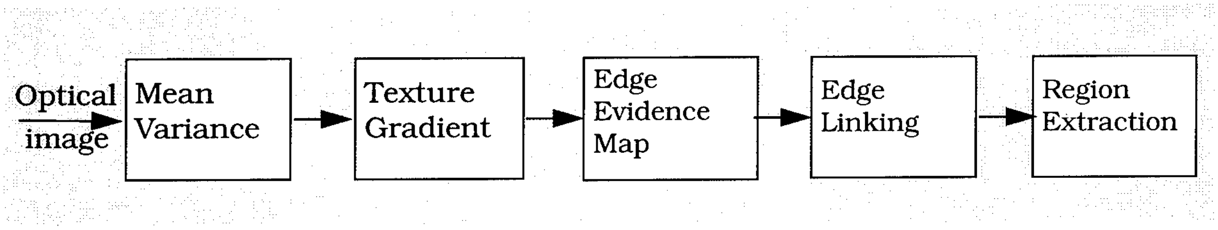

Figure 10. Functional block diagram of feature extraction

Figure 10 shows the functional block diagram of the Texture Boundary Locator kernel. The mean and variance of each pixel of the input image are first computed; they are the texture measures used to separate the regions in the image. Then the texture gradient is computed. The next step generates a multi-level edge evidence image: first the histogram of the texture gradient image, which is normalized, is computed; second, multiple thresholds are applied to produce discrete images; third, a medial axis operator is applied to the highest thresholded image producing a skeleton image; fourth, the skeleton image and the multiple thresholded images are combined together as the edge evidence image. The next block, edge linking, traces the edge evidence image linking any broken edges in the image. Lastly, the region extraction block extracts the regions and computes some basic features on each region.

13.4 Parallelization Issues

The first four blocks consist of data parallel computations. If the image is partitioned into grids, different sections of the grid may be processed in parallel. Care must be taken at the borders of the partitioned grids to prevent any border effects.

The five blocks of the feature extraction kernel must be executed in the sequence shown. However, parallel execution is possible within each block. For instance in the first block, the mean and variance can be computed in parallel, although the variance computation may need the value of the mean value. Variance computation schemes, such as computing the square of the mean and the mean square, can be applied. Then, one can compute the mean, and the mean square values in parallel.

The Texture gradient block needs to compute the directional gradients and then either selects the maximum, minimum or their average. Thus, each directional gradient can be computed in parallel. Edge evidence map combines multiple thresholded images. Once the thresholds are determined from the histogram, each threshold can be applied in parallel to the input generating the multiple thresholded images.

Edge linking also allows parallel implementation. Multiple initial edges can be traced in parallel. Care should be taken to prevent tracing the same edge on multiple processors. Although the result would remain the same; the redundant effort represents a waste of resources. The Region Extraction block behaves like edge linking. Multiple regions can be initialized and grouped in parallel. Care must be taken to prevent duplicating the same region, or else duplicated regions must first be eliminated.

13.5 Kernel Benchmark Data

We have identified and located two sets of optical images for the benchmark data. The data was taken from an aircraft flying over an airport area. Most images are 2048 x 2048; some are 1024 x 1024. This set of images were used in a Machine Learning program under ARPA/ORD sponsorship. Another optical image that can be used as benchmark data is an aerial photograph of a site for autonomous land vehicle (ALV) close to Denver, Colorado. This image is 1394 x 850 pixels. SAR image is another type of input that this kernel processes. The SAR data that was identified in other kernels, such as SAR image formation (section 6 on page 6), and map-image correlation (section 7 on page 8), can be used in this kernel

14 Decision Support Systems

14.1 Importance to C^3I Systems

Decision support systems assist the human command and control function in developing and evaluating decision options, and in distributing these decisions to the appropriate channels. In the modern battlefield, such systems are essential for commanders to rapidly access the varied sources of information and to select appropriate actions in a timely manner. Most decision support systems consist of information presentation and information dissemination systems, and thus have little computational algorithmic content. However, recent systems such as the Advanced Planning System (APS) for support of the Air Battle Planning Process and Air Tasking Order generation include capabilities for generating and evaluating options in the areas of target prioritizing, weapon selection, resource assignment and route planning.

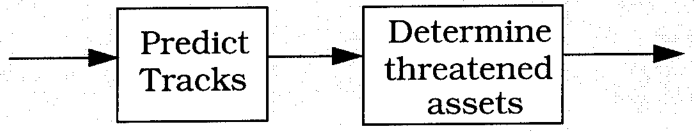

Figure 11. Illustration of Decision Support System Kernel Computations.

14.2 Kernel Approach

Our proposed kernel in decision support systems is derived from ALPHATECH's Look Ahead Battle Planner (LABP) [32] algorithms, sponsored by the Air Force's Electronic Systems Division. The LABP is a decision aid for missile defense operations which allows a commander to evaluate different rules of engagement and different resource dispositions for a prescribed attack, by providing predictions of outcomes. One of the key modules in the LABP is a target prioritizing module, which must project incoming threat trajectories and their uncertainties onto a set of defended assets, in order to determine which assets are under attack by each threat, and thus assign a value to the defense against such a threat based on their potential lethality.

Figure 11 shows the two key modules in the kernel. The inputs consist of a set of defended asset locations and a set of threat tracks with uncertain positions and velocities. The kernel must first extrapolate all trajectories and their uncertainties to impact using ballistic orbit models. Subsequently, the predicted impact points and their uncertainties are compared with the defended asset database to assign a likelihood that each asset will be destroyed. The computations require identifying all of the assets within a certain volume centered at the predicted impact point, and computing the lethality estimates.

The output of the kernel will be the lethalities assigned for each asset-threat pair. The kernel computations will be validated by establishing that the computed lethalities are within roundoff error of the standard ones obtained from a sequential implementation.

14.3 Kernel Benchmark Data

For this kernel, we propose to use the same unclassified Middle East attack scenarios which was discussed in the Threat Analysis kernel in section 12 on page 18. This will allow the ease of evaluation of different kernels on multiple architectures.

14.4 Parallelization Issues

The key computation which is hard to parallelize is the comparison of the predicted impact points with the defended asset database. In particular, the geometric nature of this problem is well-suited to using spatial data structure representations such as quad-trees or multi-grid partitions in order to obtain more efficient parallelization. We expect that speedup proportional to the number of threats is easy to obtain; for further effectiveness, partitioning of both threats and defended assets is necessary, as well as effective load balance in that case.

15 References

1. D.H. Bailey, J. Barton, T. Lasinski, and H. Simon, "The NAS Parallel Benchmarks", NASA Technical Memorandum 103863, NASA Ames Research Center, July 1993.
2. R. Hockney and M. Berry, "Public International Benchmarks for Parallel Computers", PARK-BENCH Committee: Report 1, February 7, 1994.
3. O. Drummond, D.A. Castañon and M.S. Bellovin, "Comparison of 2-D Assignment Algorithms for Rectangular, Floating Point Cost Matrices," *Proc. of SDI Tracking Workshop,* No. 4, Dec. 1990, pp. 4.81 - 4.97.
4. K.P. Dunn, "A Benchmark Scenario for SDI Tracking Algorithms," *Proc. SDI Tracking Workshop,* Huntsville, AL, July 1991.
5. T.G. Allen, L.G. Feinberg, R.O. LaMaire, K.R. Pattipati, H. Tsaknakis, R.B. Washburn, W. Wren, T. Dobbins, and P. Patterson, "Multiple Information Set Tracking Correlator (MISTC)", TR-406, Final Report, ALPHATECH, Inc., Burlington, MA, Sept., 1988.
6. Tracking Algorithm Description, ALPHATECH Report TR-615, December 1993.
7. T.G. Allen, G. Cybenko, C.M. Angelli, and J. Polito, "Hypercube Implementation of Tracking Algorithms," *Proceedings of the 1987 Command and Control Research Symposium,* Washington, D.C., 16-18 June 1987, pp. 145-153.
8. T.G. Allen, T. Kurien, and R.B. Washburn, "Parallel Computational Structures for Multiobject Tracking Algorithms on Associative Processors," *Proceedings of the 1986 American Control Conference,* Vol. 3, Seattle, Washington, 18-20 June 1986, pp. 1869-1875.
9. T.G. Allen, "Multiple Hypothesis Tracking Algorithms for Massively Parallel Computers," *Signal and Data Processing of Small Targets,* O.E. Drummond (ed.), SPIE 1698, Orlando, Florida, 1992, pp. 442-456.
10. D.P. Bertsekas and D.A. Castañon, "Parallel Synchronous and Asynchronous Implementations of the Auction Algorithm," *Parallel Computing,* Vol. 17, 1991, pp. 707-732.
11. D.P. Bertsekas and D.A. Castañon, "Parallel Asynchronous Hungarian Methods for the Assignment Problem," *ORSA J. on Computing,* Vol. 5, 1993.
12. D.P. Bertsekas, D.A. Castañon, J. Eckstein, and S. Zenios, "Parallel Computing in Network Optimization," to appear in *Handbook of Operations Research.*
13. D.A. Castañon, B. Smith and A. Wilson, "Parallel Algorithms for Assignment Problems," *SIAM Conf. on Optimization,* Boston, MA, April, 1989.
14. D.A. Castañon, "Parallel Assignment Algorithms for Multi-Object Tracking," *SDI Tracking Workshop,* Washington DC, February 1990.

15. J.C. Henry, "The Lincoln Laboratory 35 GHz Airborne Polarimetric SAR Imaging System." *IEEE National Telesystems Conference,* Atlanta, GA, 26-27 March, 1991, p.353

16. B. Zuerndorfer and G.A. Shaw, "SAR Processing for RASSP Application", Lincoln Laboratory Report.

17. G. Franceschetti, A. Mazzeo, N. Mazzocca, V. Pascazio and G. Schirinzi, "An Efficient SAR Parallel Processor," *IEEE Trans. AES,* V. 27, No. 2, March 1991.

18. L.H. Jamieson, P.T. Muller and H.J. Siegel, "FFT algorithm for SIMD parallel processing systems," *Journal of Parallel and Distributed Computing,* V. 3, 1986, pp 48-71.

19. A. Norton and J.J. Silberberger, "Parallelization and performance analysis of the Cooley-Tukey FFT algorithm for shared-memory architectures," *IEEE Trans. Computers,* V. C-5, 1987, pp 581-591.

20. L.A. Bessette, R.M. Barnes, S.C. Crocker, C.E. Frost and L.E. Guthrie, "Stockbridge Mission 85 Pass 5 Data Package (Air Calibration Site)", Lincoln Laboratory Project Report TT-80, 8 May, 1991.

21. T.A. Clark, "Analysis of the Force Level Automated Planning System (FLAPS)," RADC Report RADC-TR-89-300, October 1989.

22. Threat Avoidance Analysis for the Quiet Knight Data Processor, Honeywell Defense Avionics Systems Report A03700-MISC-RPT-00.

23. J.G. Wohl, "Force Management Decision Requirements for Air Force Tactical Command and Control," *IEEE Trans. Systems, Man & Cybernetics,* pp. 618-629, 1981.

24. K.E. Moore, M.A. Gerber, S.L. Stutsman and D.A. Brager, "Air Defense Network (ADNET) Tool for Command, Control and Communications Analysis using Petri Nets and Modeler," *Proc. 1992 Summer Computer Simulation Conference,* Reno, NV, 1992.

25. A.L. Blitz, K.E. Moore and P.A. Vail "A General Purpose Simulation Tool Exploiting the Petri Net Paradigm," *Proc. 1988 Eastern Simulation Conference: Tools for the Simulationist,* Orlando, FL, 1988.

26. K.E. Moore and R.R. Tenney, *Timing and Workload Model of the Air Operations Center: Air Tasking Order Generation and Dissemination,* SBIR Phase I Final Report. Technical Report, TR-580. Burlington, MA: ALPHATECH, Inc., April, 1993.

27. M.E. Liggins, T.G. Allen, M.A. Gerber, T.A. Kurien, and R.B. Washburn, "Multispectral Multisensor Fusion Program: Processor Load Evaluation and Opportunities for Parallel Computing," *Data Fusion Symposium,* Laurel MD, 1989, pp. 405-411.

28. Kurien, T., "Issues in the Design of Practical Multitarget Tracking Algorithms," Ch. 3 of Multitarget-Multisensor Tracking, Y. Bar-Shalom (ed.), Artech House, Norwood, MA, 1990, pp. 43-83.

29. D.A. Castañon, *Battle Management Algorithm Description Document,* ALPHATECH report TR-610, November 1993.

30. G. Cybenko and T.G. Allen, "Multidimensional Binary Partitions: Distributed Data Structures for Spatial Partitioning," *International Journal of Control,* Vol. 54, No. 6, 1991, pp. 1335-1352.

31. J. Boris, "A Vectorized Near Neighbors Algorithm of Order N Using a Monotonic Logical Grid," *Journal of Computational Physics,* Vol. 66, 1986, pp. 1-20.

32. J.J. Shaw, M.A. Gerber, R.M. James, and A. Rizzuto, "The Look-Ahead Battle Planner," *Proceedings of the American Defense Preparedness Symposium on BMD BM/C3,* Colorado Springs, March, 1994, pp. 141-146.

MISSION
OF
ROME LABORATORY

Mission. The mission of Rome Laboratory is to advance the science and technologies of command, control, communications and intelligence and to transition them into systems to meet customer needs. To achieve this, Rome Lab:

a. Conducts vigorous research, development and test programs in all applicable technologies;

b. Transitions technology to current and future systems to improve operational capability, readiness, and supportability;

c. Provides a full range of technical support to Air Force Materiel Command product centers and other Air Force organizations;

d. Promotes transfer of technology to the private sector;

e. Maintains leading edge technological expertise in the areas of surveillance, communications, command and control, intelligence, reliability science, electro-magnetic technology, photonics, signal processing, and computational science.

The thrust areas of technical competence include: Surveillance, Communications, Command and Control, Intelligence, Signal Processing, Computer Science and Technology, Electromagnetic Technology, Photonics and Reliability Sciences.